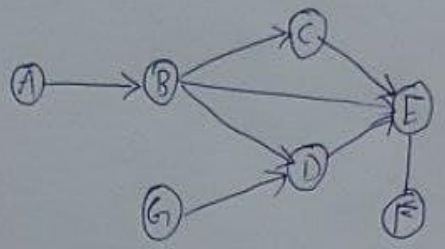
DATA STRUCTURES LAB

1) Consider a directed acyclic graph G given in the following figure



Develop a program to implement topological sorting

Algorithm

Step 1: Start

Step 2: Initialize the variables

Step 3: Input the no. of vertices

Step 4: Enter the adjacency matrix of the given graph using a for loop

Step 5: Initialize indeg[i]=0 and flag[i]=0

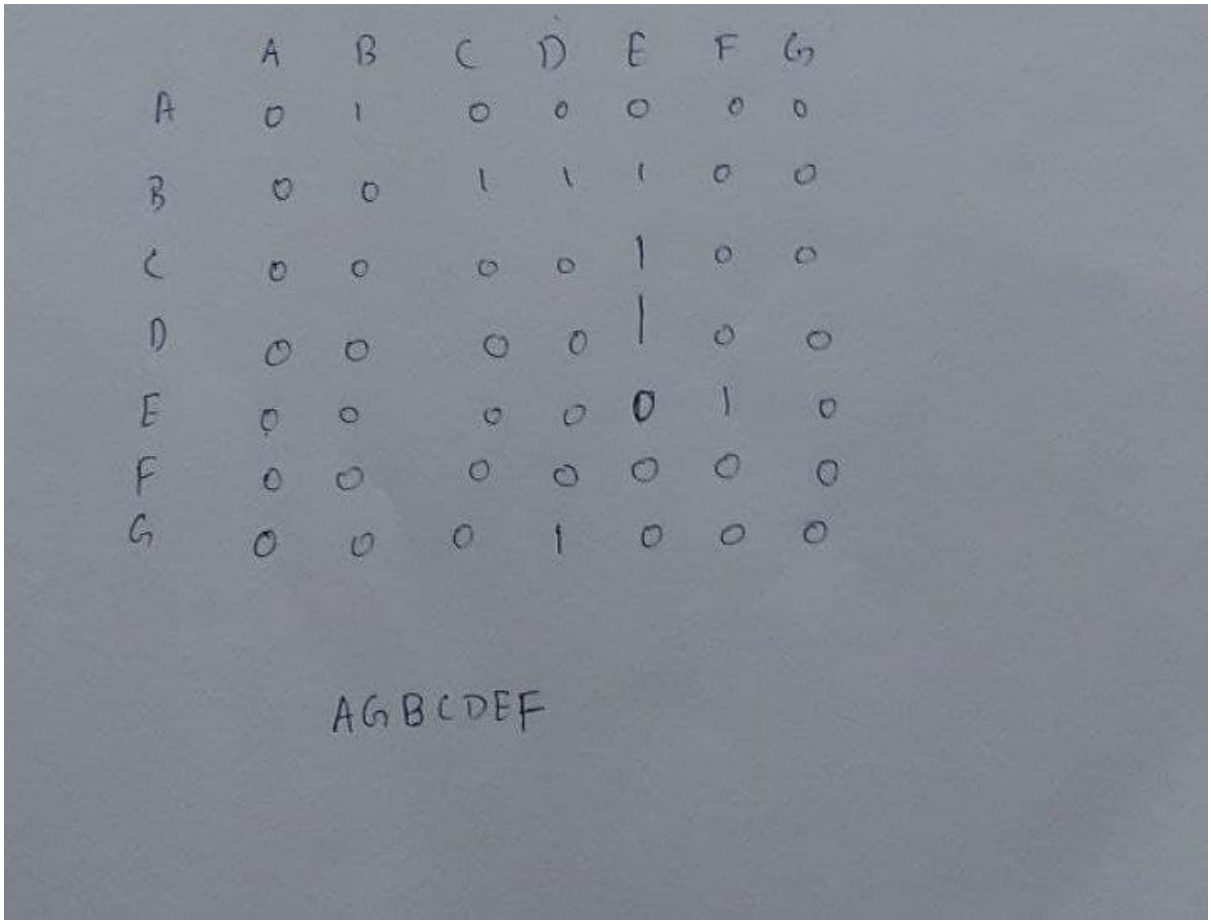
step 6: Perform topological sorting from the 1st node

step 7: Then increment flag[k]=1 and decrement indeg[k]--

Step 8: Repeat the above step to obtain topological sorting of the graph

Step 9: Print the result

Step 10: Stop


A B C D E F G
A 0 1 0 0 0 0 0
B 0 0 1 1 1 0 0
C 0 0 0 0 1 0 0
D 0 0 0 0 1 0 0
E 0 0 0 0 0 1 0
F 0 0 0 0 0 0 0
G 0 0 0 1 0 0 0
AGBCDEF

Code

```
#include<stdio.h>
int main()
{
    int i,j,k,n,a[10][10],indeg[10],flag[10],count=0;
    printf("Enter the no of vertices:\n");
    scanf("%d",&n);
    printf("Enter the adjacency matrix:\n");
    for(i=0;i<n;i++)
    {
        printf("Enter row %d\n",i+1);
        for(j=0;j<n;j++)
            scanf("%d",&a[i][j]);
    }
    for(i=0;i<n;i++)
    {
        indeg[i]=0;
        flag[i]=0;
    }
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            indeg[i]=indeg[i]+a[j][i];
    printf("\nThe topological order is:");
    while(count<n)
    {
        for(k=0;k<n;k++)
        {
            if((indeg[k]==0) && (flag[k]==0))
            {
                switch(k+1)
                {
                    case 1:printf("A");
                           break;
                    case 2:printf("B");
                           break;
                    case 3:printf("C");
                           break;
                    case 4:printf("D");
                           break;
                    case 5:printf("E");
                           break;
                    case 6:printf("F");
                           break;
                    case 7:printf("G");
                           break;
                }
                flag [k]=1;
```

```
            }
            for(i=0;i<n;i++)
            {
                if(a[i][k]==1)
                    indeg[k]--;
            }
        }
        count++;
    }
    return 0;
}
```

Output

```
Enter the no of vertices:
7
Enter the adjacency matrix:
Enter row 1
0 1 0 0 0 0 0
Enter row 2
0 0 1 1 1 0 0
Enter row 3
0 0 0 0 1 0 0
Enter row 4
0 0 0 0 1 0 0
Enter row 5
0 0 0 0 0 1 0
Enter row 6
0 0 0 0 0 0 0
Enter row 7
0 0 0 1 0 0 0

The topological order is:AGBCDEF
--------------------------------
Process exited after 116.5 seconds with return value 0
Press any key to continue . . .
```

2) Write a program for creating Doubly LL and perform the following operations

A) Insert an element at a particular position

B) Search an element

C) Delete an element at the end of the list

Algorithm

Step 1 : Start

Step 2 : Create a new node

Step 3 : Enter the choice to be inserted

Step 4 : Read n

Step 5 : Call the particular functions

Step 6 :- Create necessary functions insert_beginning(), insert_position(), delete_last(), display(), search()

Step 7 : Print the result

Step 8 : Stop

CODE

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    struct node *prev;
    struct node *next;
    int data;
};
struct node *head;
void insert_beginning();
void insert_position();
void delete_last();
void display();
void search();
void main()
{
    int choice = 0;
    while (choice != 6)
    {
        printf("\n1.Insert at beginning\n2.Insert at particular location\n3.De
lete from last\n4.Search\n5.Display\n6.Exit\n");
        printf("\nEnter your choice? = ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1: insert_beginning();
                break;
        case 2: insert_position();
                break;
        case 3: delete_last();
                break;
        case 4: search();
                break;
        case 5: display();
                break;
        case 6: exit(0);
                break;
        default:printf("Please enter a valid choice");
        }
    }
}
void insert_beginning()
{
    struct node *ptr;
```

```
    int item;
    ptr = (struct node *)malloc(sizeof(struct node));
    if (ptr == NULL)
    {
        printf("\nOVERFLOW");
    }
    else
    {
        printf("Enter Item value = ");
        scanf("%d", &item);

        if (head == NULL)
        {
            ptr->next = NULL;
            ptr->prev = NULL;
            ptr->data = item;
            head = ptr;
        }
        else
        {
            ptr->data = item;
            ptr->prev = NULL;
            ptr->next = head;
            head->prev = ptr;
            head = ptr;
        }
        printf("Node inserted");
    }
}

void insert_position()
{
    struct node *ptr, *temp;
    int item, loc, i;
    ptr = (struct node *)malloc(sizeof(struct node));
    if (ptr == NULL)
    {
        printf("\n OVERFLOW");
    }
    else
    {
        temp = head;
        printf("Enter the location = ");
        scanf("%d", &loc);
        for (i = 0; i < loc-1; i++)
        {
            temp = temp->next;
            if (temp == NULL)
```

```
            {
                printf("\n There are less than %d elements", loc);
                return;
            }
        }
        printf("Enter value = ");
        scanf("%d", &item);
        ptr->data = item;
        ptr->next = temp->next;
        ptr->prev = temp;
        temp->next = ptr;
        temp->next->prev = ptr;
        printf("\nnode inserted\n");
    }
}

void delete_last()
{
    struct node *ptr;
    if (head == NULL)
    {
        printf("\n UNDERFLOW");
    }
    else if (head->next == NULL)
    {
        head = NULL;
        free(head);
        printf("\nnode deleted");
    }
    else
    {
        ptr = head;
        while (ptr->next != NULL)
        {
            ptr = ptr->next;
        }
        ptr->prev->next = NULL;
        free(ptr);
        printf("\nnode deleted");
    }
}
void display()
{
    struct node *ptr;
    printf("\nValues-\n");
    ptr = head;
    while (ptr != NULL)
    {
```

```
        printf("%d\n", ptr->data);
        ptr = ptr->next;
    }
}
void search()
{
    struct node *ptr;
    int item, i = 0, flag;
    ptr = head;
    if (ptr == NULL)
    {
        printf("\nEmpty List");
    }
    else
    {
        printf("\nEnter item which you want to search?");
        scanf("%d", &item);
        while (ptr != NULL)
        {
            if (ptr->data == item)
            {
                printf("\nitem found at location %d ", i );
                flag = 0;
                break;
            }
            else
            {
                flag = 1;
            }
            i++;
            ptr = ptr->next;
        }
        if (flag == 1)
        {
            printf("\nItem not found");
        }
    }
}
```

OUTPUT

```
1.Insert at beginning
2.Insert at particular location
3.Delete from last
4.Search
5.Display
6.Exit

Enter your choice? = 1
Enter Item value = 10
Node inserted
1.Insert at beginning
2.Insert at particular location
3.Delete from last
4.Search
5.Display
6.Exit

Enter your choice? = 1
Enter Item value = 20
Node inserted
1.Insert at beginning
2.Insert at particular location
3.Delete from last
4.Search
5.Display
6.Exit

Enter your choice? = 1
Enter Item value = 30
Node inserted
1.Insert at beginning
2.Insert at particular location
3.Delete from last
4.Search
5.Display
6.Exit

Enter your choice? = 1
Enter Item value = 40
Node inserted
1.Insert at beginning
2.Insert at particular location
3.Delete from last
4.Search
5.Display
6.Exit

Enter your choice? = 1
Enter Item value = 50
Node inserted
1.Insert at beginning
2.Insert at particular location
3.Delete from last
4.Search
5.Display
6.Exit

Enter your choice? = 2
Enter the location = 3
Enter value = 25

node inserted
```

```
1.Insert at beginning
2.Insert at particular location
3.Delete from last
4.Search
5.Display
6.Exit

Enter your choice? = 5

Values-
50
40
30
25
20
10

1.Insert at beginning
2.Insert at particular location
3.Delete from last
4.Search
5.Display
6.Exit

Enter your choice? = 3

node deleted
1.Insert at beginning
2.Insert at particular location
3.Delete from last
4.Search
5.Display
6.Exit

Enter your choice? = 5

Values-
50
40
30
25
20

1.Insert at beginning
2.Insert at particular location
3.Delete from last
4.Search
5.Display
6.Exit

Enter your choice? = 4

Enter item which you want to search?20

item found at location 4
1.Insert at beginning
2.Insert at particular location
3.Delete from last
4.Search
5.Display
6.Exit
```

Git link:-
https://github.com/NithinNitz12/DataStructures/tree/master/LAB%20EXAM